Proceedings of the International Conference on Industrial Engineering and Operations Management
Bangkok, Thailand, March 5-7, 2019

Designing Supervisory Policies that Avoid Livelocks in Manufacturing- and Service-Systems Modeled using General Petri Nets: A Tutorial using an Illustrative Example

R. Khalegi, A. Raman and R.S. Sreenivas
Department of Industrial and Enterprise Systems Engineering
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
khalegh2@illinois.edu, raman12@illinois.edu, rsree@illinois.edu,

Abstract

When a Manufacturing- or Service-System gets into a *deadlocked* state *none* of the activities can proceed towards completion. When the same system is in a *livelocked* state, some activity of the system enters a state of suspended animation and is never completed, while other activities proceed to completion without hindrance. A livelock-free system does not experience deadlocks; but a deadlock-free system can experience livelocks. This tutorial is about the synthesis of livelock avoidance policies for Manufacturing- and Service-Systems that are modeled by *Petri Nets* (PNs).

After characterizing *livelock-freedom*, we introduce the paradigm of *liveness enforcing supervisory policies* (LESPs). An LESP is said to be *minimally restrictive*, if the fact that it prevents the occurrence of an event at a given state is reason enough to infer that *all* livelock-avoidance policies would do the same. If there is an LESP for a system, there is a unique minimally restrictive LESP. We restrict attention to LESPs that can be implemented using a *Disjunctive Normal Form* (DNF) expression on the state of the PN modeling the system. We describe the process of synthesizing the minimally restrictive LESPs using an illustrative example using a software-product developed by the authors.

Keywords

Manufacturing Systems, Service Systems, Supervisory Control, Livelock Avoidance

1. Introduction

The domain of discourse for this paper is the operational level control of a manufacturing system, the contention-resolution of shared resources in computer networks, or the operations-management of multi-component organizations with event driven dynamics like shipyards and airports, etc. These systems are representatives of the family of *Discrete-Event/Discrete-State* (DEDS) systems. They are characterized by state variables that have a logical or symbolic interpretation. The dynamics of DEDS systems arise due to the occurrence of specific events, and the state-variables of such systems are piecewise constant with discontinuities at the occurrence-instants of these events. At each discrete-state, there are potential discrete-events that can occur, the occurrence of any one of them would change the state of the system, which then results in a new set of events that could occur, and this process can be repeated as often as necessary. Air-traffic control systems; automated manufacturing systems; computer networks; integrated command, control, communication and information (C^3I) systems; operations-management of multi-component organizations with event-driven dynamics like shipyards, airports, hospitals, etc. are examples

© IEOM Society International

of DEDS systems.

DEDS systems are regulated by a *supervisory policy*, which determines which event is to be permitted at each state, in such a manner that some behavioral specification is satisfied. Our focus is on the synthesis and implementation of *liveness enforcing supervisory policies* (LESPs). A DEDS system is *live* (Alpern and Schneider, 1985), if irrespective of the past, every event can be executed, not necessarily immediately, in the future. A live DEDS system does not experience *livelocks*, and serves as the main motivation for investigations into the synthesis and implementation of LESPs. A DEDS system is in a livelocked-state, if some event has entered into a state of suspended animation for perpetuity. If every event of the DEDS system is in a state of suspended animation, the DEDS system is *deadlocked*. A livelock-free DEDS system does not have deadlocked-states, but a deadlock-free DEDS system can still experience livelocks. Livelock freedom is harder to achieve, compared to deadlock freedom.

Petri nets (PNs) are a popular modeling paradigm for DEDS systems with asynchronous events (Peterson 1981; Murata 1989). In this modeling framework, each task is represented by a *transition*, and a *marking* represents the state of the system. For each marking of the PN, there is a set of possible transitions that can *fire* (i.e. a set of events that can occur). The *firing* of the transition results in a new marking, and the process repeats as often as necessary. For a PN model that is initialized with a marking, there is a set of *reachable markings* that can be realized through the process described above. A PN is said to be *live*, if it is possible to fire any transition (although not immediately) from every reachable marking. We consider PN models that are not live, but can be made live under the supervision of an LESP. The synthesis of the LESP is essentially a prescription for livelock avoidance in Manufacturing- and Service-Systems modeled by a PN. These concepts are formally introduced and developed in the next section.

The rest of the paper is organized as follows – section 2 presents the notations and definitions used in this paper. Section 3 describes the process of synthesizing PN models for manufacturing- and service-systems. Following this, in section 4 we explain the paradigm of supervisory control of DEDS systems. We present an illustrative example of an automated painting booth that has to be supervised to prevent the occurrence of deadlocks.

2. Notations and Definitions

$\mathbb{N}$ ($\mathbb{N}^+$) denotes the set of non-negative (positive) integers. The cardinality of a set A is represented as *card(A)*. A *Petri net* (PN) structure $N = (\Pi, T, \Phi, \Gamma)$ is an ordered 4-tuple, where $\Pi = \{p_1 \cdots p_n\}$ is a set of *n places*, $T = \{t_1 \cdots t_m\}$ is a collection of *m transitions*, $\Phi \subseteq (\Pi \times T) \cup (T \times \Pi)$ is a set of *arcs*, and $\Gamma: \Phi \to \mathbb{N}^+$ is the *weight* associated with each arc. The weight of an arc is represented by an integer that is placed alongside the arc. If an arc has a unitary weight, it is not represented in its graphical representation in this paper.

If all arcs of a PN are unitary, it is said to be an *ordinary* PN, otherwise it is a *general* PN. The *initial marking* of a PN structure N is a function $\boldsymbol{m}^0: \Pi \to \mathbb{N}$, which identifies the number of tokens in each place. A *Petri net* (PN), $N(\mathbf{m}^0)$, is a PN structure N together with its initial marking $\mathbf{m}^0$.

T^* represents the set of all finite-length strings of transitions. For $\sigma \in T^*$, we use $\boldsymbol{x}(\sigma)$ to denote the *Parikh vector* of σ. That is, the i^{th} entry, $\boldsymbol{x_i}(\sigma)$, corresponds to the number of occurrences of transition t_i in σ.

Let ${}^\bullet x \coloneqq \{y | (y,x) \in \Phi\}$ and $x^\bullet \coloneqq \{y | (x,y) \in \Phi\}$. If $\forall p \in {}^\bullet t, \boldsymbol{m}^i(p) \geq \Gamma((p,t))$ for some $t \in T$ and some marking $\mathbf{m}^i$, then $t \in T$ is said to be enabled at marking $\mathbf{m}^i$. The set of enabled transitions at marking $\mathbf{m}^i$ is denoted by the symbol $T_e(N, \boldsymbol{m}^i)$. An enabled transition $t \in T_e(N, \boldsymbol{m}^i)$ can fire, which changes the marking $\mathbf{m}^i$ to $\mathbf{m}^{i+1}$ according to $\boldsymbol{m}^{i+1}(p) = \boldsymbol{m}^i(p) - \Gamma(p,t) + \Gamma(t,p)$.

© IEOM Society International

A set of markings $\mathcal{M} \subseteq \mathbb{N}^n$ is said to be *right-closed* if $((\boldsymbol{m}^1 \in \mathcal{M}) \wedge (\boldsymbol{m}^2 \geq \boldsymbol{m}^1) \Rightarrow (\boldsymbol{m}^2 \in \mathcal{M}))$, and is uniquely defined by its finite set of minimal-elements.

When the marking is interpreted as a nonnegative integer-valued vector, it is useful to define the input matrix **IN** (output matrix **OUT**) as an $n \times m$ matrix, where $\mathbf{IN}_{i,j}$ ($\mathbf{OUT}_{i,j}$) equals $\Gamma((p_i,t_j))$ ($\Gamma((p_i,t_j))$) if $p_i \in {}^\bullet t_j$, ($p_i \in t_j^\bullet$) and is zero-valued otherwise. The *incidence matrix* **C** of the PN N is an $n \times m$ matrix, where **C** = **OUT**−**IN**.

3. Petri Net Models of Manufacturing- and Service-Systems

The manufacturing process for discrete parts manufacture, automated assembly, and service-systems can be decomposed into a set of distinct operations, where each operation represents a single stage that is executed independently of others. These operations might have precedence relations that impose a partial ordering on how these operations can be executed over time. Typically, each operation requires a set of resources, such as raw materials, fixtures and specialized machinery. A set of integer-valued resource states is associated with each resource that captures the conditions required to execute each operations. For example, a binary resource state associated with raw material could indicate if material is available, or not available, for processing. The number of customers waiting to be served in a service-system could be represented by an unbounded, integer-valued set.

The operations of the system can commence only if some of the resource states are no less than some value. For example, a server can commence a service-task only if there is at least one customer that is waiting to be served. Upon completion, the operations will change the resource state, and the process can continue as often as necessary.

To build a PN model of a manufacturing- or service-system, the discrete resource states are mapped to places. The number of tokens associated with the place corresponds to the value of the resource state. Transitions are used to represent operations. The input arcs to the transition, and their associated weight, capture the requirements on the resources that are a prerequisite to the commencement of the operation. The output arcs from the transition, and their associated weights, capture the changes to the resource states that would result after the operation represented by the transition is completed. We refer the reader to Chapter 2 of (Hack 1972), or (Girault and Valk, 2003) for additional information about building PN models for manufacturing- and service-systems.

4. Supervisory Control of PNs

Under this paradigm, the set of transitions in the PN is partitioned into a set of *controllable transitions* ($T_c \subseteq T$) and a set of *uncontrollable transitions* ($T_u \subseteq T$). The controllable (uncontrollable) transitions are represented as filled (unfilled) boxes in graphical representation of PNs.

A supervisory policy $\wp: \mathbb{N}^n \times T \rightarrow \{0,1\}$, is a function that returns a 0 or 1 for each transition and each reachable marking. Transition $t \in T$ is *control-enabled* (*state-enabled*) if $\wp(\boldsymbol{m}, t) = 1$ ($t \in T_e(N, \boldsymbol{m})$) for some marking **m**. A transition that is state- and control-enabled can fire, which results in a new marking as indicated in the previous section. Since uncontrollable transitions cannot be prevented from firing by the supervisory policy, we require the following condition to be true of all supervisory policies: $\forall \boldsymbol{m} \in \mathbb{N}^n, \wp(\boldsymbol{m}, t) = 1$, if $t \in T_u$.

A *valid firing string* $\sigma = t_1 t_2 \cdots t_k \in T^*$ for a marking $\mathbf{m}^i$ satisfies the following conditions: (1) $t_1 \in T_e(N, \mathbf{m}^i), \wp(\mathbf{m}^i, t_1) = 1$, and (2) for $j \in \{1,2, \cdots k-1\}$ the firing of transition t_j produces a marking $\mathbf{m}^{i+j}$, $t_{j+1} \in T_e(N, \mathbf{m}^{i+j})$, and $\wp(\mathbf{m}^{i+j}, t_{j+1}) = 1$.

© IEOM Society International

$\Re(N, \boldsymbol{m}^0, \wp)$ denotes the set of markings that are reachable from $\mathbf{m}^0$ under the supervision of $\wp$ in *N*. We use $\boldsymbol{m}^i \xrightarrow{\sigma} \boldsymbol{m}^j$ to denote that $\mathbf{m}^j$ results from the firing of $\sigma \in T^*$ from $\mathbf{m}^i$.

A transition t_k is *live* under the supervision of $\wp$ if $\forall \boldsymbol{m}^i \in \Re(N, \boldsymbol{m}^0, \wp), \exists \boldsymbol{m}^j \in \Re(N, \boldsymbol{m}^i, \wp)$ such that $t_k \in T_e(N, \mathbf{m}^j)$ and $\wp(\mathbf{m}^j, t_k) = 1$. If all transitions in $N(\mathbf{m}^0)$ are live under $\wp$, then it is a *liveness enforcing supervisory policy* (LESP) for $N(\mathbf{m}^0)$. The policy $\wp$ is said to be *minimally restrictive* if for every LESP $\widehat{\wp}: \mathbb{N}^n \times T \to \{0,1\}$, the following condition holds $\forall \boldsymbol{m}^i \in \mathbb{N}^n, \forall\, t \in T, \wp(\mathbf{m}^i, t) \geq \widehat{\wp}(\mathbf{m}^i, t)$.

There is an LESP for $N(\mathbf{m}^0)$ if and only if $\boldsymbol{m}^0 \in \Delta(\boldsymbol{N})$, where $\Delta(N) = \{\boldsymbol{m}^0 \in \mathbb{N}^{card(\Pi)} | \exists\ an\ LESP\ for\ N(\boldsymbol{m}^0)\}$ is the set of initial markings $\mathbf{m}^0$ for which there is a LESP for $N(\mathbf{m}^0)$. It follows that $\Delta(N)$ is *control invariant* (cf. Ramadge and Wonham, 1987; Sreenivas 1997; Sreenivas 2012) with respect to *N*; that is, if $\boldsymbol{m}^1 \in \Delta(N), t_u \in T_u \cap T_e(N, \boldsymbol{m}^1)$ and $\boldsymbol{m}^1 \xrightarrow{t_u} \boldsymbol{m}^2$ in *N*, then $\boldsymbol{m}^2 \in \Delta(N)$. Equivalently, only the firing of a controllable transition at any marking in $\Delta(N)$ can result in a new marking that is not in $\Delta(N)$.

If $\wp$ is an LESP for $N(\mathbf{m}^0)$, then $\Re(N, \boldsymbol{m}^0, \wp) \subseteq \Delta(N)$. Additionally, the LESP $\wp^*$, that prevents the firing of a controllable transition at any marking when its firing would result in a new marking that is not in $\Delta(\boldsymbol{N})$, is the *minimally restrictive* LESP for $N(\mathbf{m}^0)$. That is, there can be no other LESP, independent of the implementation paradigm, that can be better than $\wp^*$.

Neither the existence, nor the non-existence, of an LESP for an arbitrary PN is semi-decidable; the existence of an LESP is decidable if all transitions in the PN are controllable, or if the PN structure belongs to specific classes identified in the literature (Sreenivas 1997; Sreenivas 2012; Somnath and Sreenivas 2013; Sreenivas 2013; Salimi, Somnath and Sreenivas 2015). The process of deciding the existence of an LESP in an arbitrary instance from these classes is NP-hard.

If $\Delta(N)$ is known to be right-closed for a PN structure *N*, then the software described in (Chandrasekaran, Somnath and Sreenivas, 2015; Salimi, Somnath and Sreenivas, 2015) can be used to synthesize the minimally restrictive LESP for $N(\mathbf{m}^0)$ if $\mathbf{m}^0 \in \Delta(N)$. To date, the $\mathcal{H}$ class of PN structures is the largest class of PN structures for which the set $\Delta(N)$ is known to be right-closed. A PN structure $N = (\Pi, T, \Phi, \Gamma)$ belongs to the class $\mathcal{H}$ if and only if $\forall p \in \Pi, \forall t_u \in p^\bullet \cap T_u$,

$$\left(\Gamma(p, t_u) = \min_{t \in p^\bullet} \Gamma(p, t)\right) \wedge (\forall t \in \Omega(t_u), {}^\bullet t_u \subseteq {}^\bullet t),$$

where $\Omega(t) = \{\hat{t} \in T | {}^\bullet t \cap {}^\bullet \hat{t} \neq \emptyset\}$, denote the set of transitions that share a common input place with $t \in T$.

That is, for any member of the $\mathcal{H}$ class of PN structures: (1) for each place, the weights associated with the outgoing arcs that terminate on uncontrollable transitions must be the smallest of all outgoing arc-weights, and (2) the set of input places to each uncontrollable transition is no larger than the set of input places of any transition which shares a common input place with it. This class of PN structures strictly includes the class of *Fully Controlled* PNs (i.e. $T_u = \emptyset$), the class of *Ordinary Free Choice* PNs (Hack 1972; Murata 1989), and the class of PNs described in references (Somnath and Sreenivas, 2013; Sreenivas 2013). If *N* is a PN structure for which $\Delta(N)$ is right-closed, the maximally permissive LESP for $N(\mathbf{m}^0)$ if $\mathbf{m}^0 \in \Delta(N)$ can be equivalently represented by a series of *Disjunctive Normal Formulae* (DNF), $\{\Phi(t_c, \boldsymbol{m})\}_{t_c \in T_c}$, where each controllable transition $t_c \in T_c$ is control-enabled at marking $\boldsymbol{m}$ if and only if the formula $\Phi(t_c, \boldsymbol{m})$ evaluates to "*true*" (cf. Devarakonda and Sreenivas, 2015). These concepts are illustrated using an example in the next section.

© IEOM Society International

We say a policy $\wp$ is *marking-monotone* if $\wp(\mathbf{m}^i, t_1) = 1$ for some marking $\mathbf{m}^i$ implies for $\wp(\mathbf{m}^j, t_1) = 1$ for any $\boldsymbol{m}^j \geq \boldsymbol{m}^i$. In addition, we say an LESP $\wp$ for $N(\mathbf{m}^i)$ is a *marking-monotone-LESP* if $\wp$ is an LESP for $N(\mathbf{m}^j)$ for any $\boldsymbol{m}^j \geq \boldsymbol{m}^i$. Given an arbitrary PN structure N, it is possible to decide if there is a marking-monotone-LESP for any $N(\mathbf{m}^0)$ (Chen, Raman, Hu and Sreenivas, 2018).

If $\Delta(N)$ is known to be right-closed for a general PN N, then the minimally restrictive LESP $\wp^*$, that prevents the firing of a controllable transition at any marking when its firing would result in a new marking that is not in $\Delta(\boldsymbol{N})$, will be a *marking-monotone-LESP.*

5. An Illustrative Example

We consider an automated system that paints automobile bodies using two paint-booths (*Booth 1* and *Booth 2*), and two robots (*Robot 1* and *Robot 2*). The paint-booths have a stationary fixture where the automobile body remains till the painting tasks are completed. Robot 1 is used to transportation tasks, while Robot 2 is used for painting tasks.

A base-coat is applied to an unfinished body in Booth 1, following this it is transferred to Booth 2, where a coat of clear resin is applied. After sufficient time has elapsed for the solvents to evaporate, the finished automobile frame is transported out of Booth 2, and a new unfinished frame is placed in Booth 1 for painting.

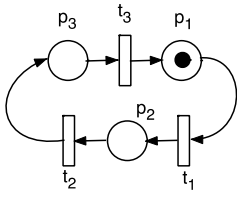

Figure 1: The PN model representing the task-flow for Robot 1.

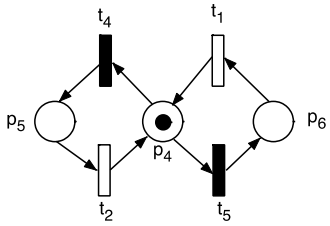

Figure 2: The PN model representing the task-flow for Robot 2.

The base-coat operation can commence only after Robot 1 has placed an unfinished automobile body in Booth 1, and the base-coat paint nozzle is affixed to Robot 2. After the completion of this task, the automobile body with the base-coat is ready to be transported to Booth 2 by Robot 1. At this stage, Robot 2 can either be directed to do another base-coat operation in Booth 1; or, it can be commissioned to commence a clear-coat operation in Booth 2.

The clear-coat operation can start only after Robot 1 has placed the body with the base-coat in Booth 2, and the clear-coat nozzle is affixed to Robot 2. After this task is completed, the finished automobile body is ready to be transported out of Booth 2 by Robot 1, which then places a new unfinished automobile body in Booth 1 for its base-coat.

Figure 1 shows the PN model representing the task sequence for Robot 1. Transition t_1 represents the operation of putting the base-coat in Booth 1. Transition t_2 represents the operation of putting the clear coat in Booth 2. Transition t_3 denotes the operation of removing the painted automobile body from Booth 2, along with the process of placing a new, unfinished body on the fixture in Booth 1. The presence of a token in place p_1 (resp. p_2) denotes the resource state fact that an automobile body is on the fixture in Booth 1 (resp. Booth 2). A token in place p_3 denotes the resource state that a finished automobile body is at the fixture in Booth 2 to be removed.

© IEOM Society International

Figure 2 shows the PN model for the task sequence for Robot 2. As noted above, transitions t_1 and t_2 represent the operations of putting the base-coat and the clear-coat in Booth 1 and Booth 2, respectively. Transition t_4 (resp. t_5) represents the operation where Robot 2 is commissioned to the clear-coat (resp. base-coat) paint-operation. The presence of a token in place p_4 indicates Robot 2 is ready to be commissioned for any of the two paint tasks. The present of a token in place p_5 (resp. place p_6) indicates Robot 2 is ready to apply the clear-coat (resp. the base-coat) in Booth 2 (resp. Booth 1). The transitions t_4 and t_5 are controllable, in that the decisions about how Robot 2 is commissioned is to be made by the supervisory policy enunciated in subsequent text.

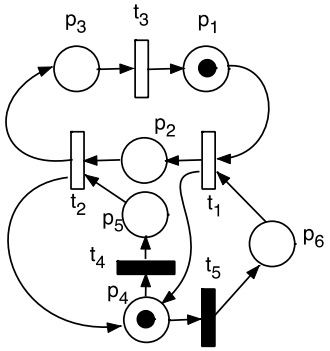

Figure 3: The PN model that results from merging the PN models shown in Figures 1 and 2.

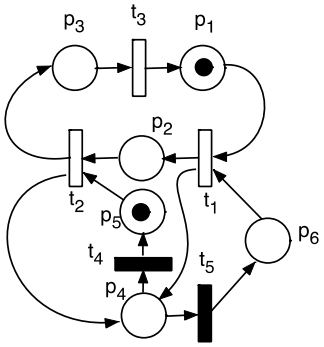

Figure 4: The Deadlock that arises from commissioning Robot 2 for a clear-coat operation while there is an automobile body waiting for its base-coat.

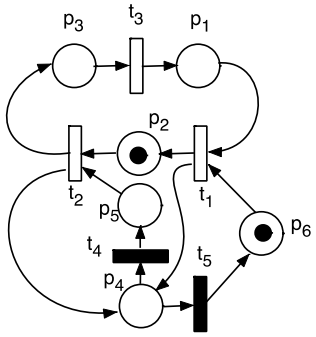

Figure 5: The Deadlock that arises from commissioning Robot 2 for a base--coat operation while there is an automobile body waiting for its clear-coat.

Figure 3 shows the PN model that results from merging the two PNs of Figures 1 and 2. Figure 4 demonstrates the fact that the improper decision of commissioning Robot 2 to the clear-coat operation, while there is an unfinished automobile body waiting for its base-coat paint in Booth 1, will result in a deadlock where no activity can be completed. Figure 5 demonstrates the deadlock that arises when Robot 2 is commissioned to the base-coat operation while there is an automobile body waiting for its clear-coat. We will now demonstrate the use of the software product described in references (Chandrasekaran, Somnath and Sreenivas 2015; Salimi, Somnath and Sreenivas 2015) to generate a livelock/deadlock avoidance policy for this PN model.

The input file for the PN shown in figure 3, as per the requirements for the software of references (Chandrasekaran, Somnath and Sreenivas 2015; Salimi, Somnath and Sreenivas 2015) is shown in figure 6. Figure 7 shows the output generated by this software tool. For the PN shown in figure 3, the set $\Delta(N)$ is identified by the minimal elements

$$\{(1\,0\,0\,0\,0\,1)^T, (0\,0\,1\,0\,0\,1)^T, (0\,1\,0\,0\,1\,0)^T, (1\,0\,0\,1\,0\,0)^T, (0\,1\,0\,1\,0\,0)^T, (0\,0\,1\,1\,0\,0)^T\}.$$

That is, for any $\mathbf{m^0} \in \Delta(N)$ (i.e. any $\mathbf{m^0} \in \mathbb{N}^6$ that is greater-than-or-equal-to any one of the six minimal elements listed above), there is a supervisory policy that can prevent the occurrence of any livelocks (and consequently, any deadlocks as well).

This policy will prevent the firing of controllable transitions t_4 (resp. t_5) at any marking $\mathbf{m^1} \in \mathbb{N}^6$ if $\boldsymbol{m}^1 \xrightarrow{t_4} \boldsymbol{m}^2$ (resp. $\boldsymbol{m}^1 \xrightarrow{t_5} \boldsymbol{m}^2$) and $\boldsymbol{m^2} \notin \Delta(N)$. This policy is the minimally restrictive LESP for $N(\mathbf{m}^0)$ for any $\mathbf{m^0} \in \Delta(N)$, where N is the PN structure shown in figure 3. Alternately, from the results of reference (Devarakonda and Sreenivas, 2015), the minimally restrictive LESP for $N(\mathbf{m}^0)$ for any $\mathbf{m^0} \in \Delta(N)$, where N is the PN structure shown in figure 3 will permit the firing of controllable transition t_4 only if this formula evaluates is true:

© IEOM Society International

$$\{((\boldsymbol{m}(p_1) \geq 1) \wedge (\boldsymbol{m}(p_6) \geq 1)) \vee ((\boldsymbol{m}(p_3) \geq 1) \wedge (\boldsymbol{m}(p_6) \geq 1)) \vee (\boldsymbol{m}(p_2) \geq 1) \vee ((\boldsymbol{m}(p_1) \geq 1) \wedge (\boldsymbol{m}(p_4) \geq 1)) \vee ((\boldsymbol{m}(p_3) \geq 1) \wedge (\boldsymbol{m}(p_4) \geq 1))\}.$$

That is, Robot 2 will be tasked to start the clear-coat operation at a marking $\boldsymbol{m}$ if and only if the above DNF evaluates to "true" at $\boldsymbol{m}$.

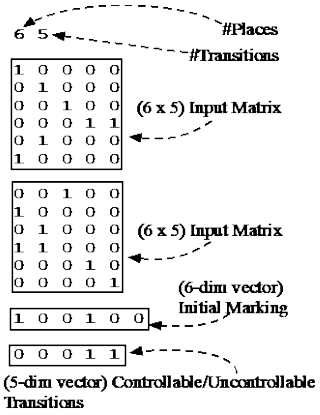

Figure 6: The input file for the PN shown in figure 3.

```
Input File = "pn1"

Incidence Matrix :

   T  1  2  3  4  5
P
  1  -1  .  1  .  .
  2   1 -1  .  .  .
  3   .  1 -1  .  .
  4   1  1  . -1 -1
  5   . -1  .  1  .
  6  -1  .  .  .  1

Initial Marking : ( 1 0 0 1 0 0 )

There is an LESP for this (fully controlled) PN

_____________________________________________


Minimal Elements of the fully controlled Net
--------------------------------------------

 1: ( 1 0 0 0 0 1 )
 2: ( 0 0 1 0 0 1 )
 3: ( 0 1 0 0 1 0 )
 4: ( 1 0 0 1 0 0 )
 5: ( 0 1 0 1 0 0 )
 6: ( 0 0 1 1 0 0 )


List of Controllable Transitions
--------------------------------
t4 t5


(Final) Minimal Elements of the control-invariant set
-----------------------------------------------------

 1: ( 1 0 0 0 0 1 )
 2: ( 0 0 1 0 0 1 )
 3: ( 0 1 0 0 1 0 )
 4: ( 1 0 0 1 0 0 )
 5: ( 0 1 0 1 0 0 )
 6: ( 0 0 1 1 0 0 )


This is An LESP
```

Figure 7: The output generated by the software of references (Chandrasekaran, Somnath and Sreenivas 2015; Salimi, Somnath and Sreenivas 2015)

Likewise, controllable transition t_5 is permitted only if this formula evaluates is true:

$$\{(\boldsymbol{m}(p_1) \geq 1) \vee (\boldsymbol{m}(p_3) \geq 1) \vee ((\boldsymbol{m}(p_2) \geq 1) \wedge (\boldsymbol{m}(p_5) \geq 1)) \vee ((\boldsymbol{m}(p_2) \geq 1) \wedge (\boldsymbol{m}(p_4) \geq 1))\}.$$

Or, Robot 2 will start the base-coat operation at a marking $\boldsymbol{m}$ if and only if the above DNF evaluates to "true" at $\boldsymbol{m}$.

6. Conclusions

Using an illustrative example we described how the operations in a manufacturing- or service-systems could be modeled using Petri nets. These systems are prone to livelocks, where an activity never progresses to completion, while the other activities proceed normally. We showed how a livelock avoidance policy could be synthesized for these systems using a software tool from the literature.

Acknowledgements

This work was supported in part by the *Arthur Davis Faculty Scholar Endow*ment at the University of Illinois at Urbana-Champaign.

© IEOM Society International

References

Alpern, B. and Schneider, F.B, "Defining liveness," *Information Processing Letters*, vol. 21, no. 4, 1985.

Chandrasekaran, S, Somnath, N. and Sreenivas, R.S., "A Software Tool for the Automatic Synthesis of Minimally Restrictive Liveness Enforcing Supervisory Policies for a class of General Petri Nets," *Journal of Intelligent Manufacturing*, Volume 26, No. 5, October, 2015, 945-958.

Chen, C, Raman, A. Hu, H and Sreenivas, R.S., "On Liveness Enforcing Supervisory Policies for Arbitrary Petri Nets," *IEEE Transactions on Automatic Control*, under review, November 2018.

Devarakonda, V. and Sreenivas, R.S., "On a Sufficient Information Structure for Supervisory Policies that Enforce Liveness in a Class of General Petri Nets," *IEEE Transactions on Automatic Control*, Vol. 60, No. 7, July 2015, 1915-1921.

Girault, C. and Valk, R., Petri Nets for Systems Engineering: A Guide to Modeling, Verification and Applications, Springer Verlag, 2003.

Hack, M.H.T., Analysis of Production Schemata by Petri Nets, Masters Thesis, Project MAC, Massachusetts Institute of Technology, Cambridge, MA, February 1972.

Murata, T., "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541 580, 1989.

Peterson, J.L. *Petri net theory and the modeling of systems*. Prentice Hall, 1981.

Ramadge. P.J.G. and Wonham, W.,"Modular feedback logic for discrete event systems," *SIAM J. Control and Optimization*, vol. 25, no. 5, pp. 1202–1218, September 1987.

Salimi, E. Somnath, N. and Sreenivas, R.S., "A Software Tool for Live-Lock Avoidance in Systems Modeled using a Class of Petri Nets," *International Journal of Computer Science, Engineering and Applications*, vol. 5, no. 2, pp. 1–13, October 2015.

Somnath, N. and Sreenivas, R.S., "On Deciding the Existence of a Liveness Enforcing Supervisory Policy in a Class of Partially-Controlled General Free-Choice Petri Nets," *IEEE Transactions on Automation Science and Engineering*, vol. 10, pp. 1157–1160, April 2015.

Sreenivas, R.S., "On the existence of supervisory policies that enforce liveness in discrete-event dynamic systems modeled by controlled Petri nets," IEEE Transactions on Automatic Control, vol. 42, no. 7, pp. 928–945, July 1997.

Sreenivas, R.S., "On the existence of supervisory policies that enforce liveness in partially controlled free-choice petri nets," *IEEE Transactions on Automatic Control*, vol. 57, no. 2, pp. 435–449, February 2012.

Sreenivas, R.S., "On a decidable class of partially controlled petri nets with liveness enforcing supervisory policies," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 5, pp. 1256–1261, August 2013.

Biographies

Roshanak Khalegi is a graduate student pursuing her Ph.D. in Industrial Engineering at the Department of Industrial and Enterprise Systems Engineering at the University of Illinois at Urbana-Champaign. She completed her Bachelor of Science and Master of Science degrees in Industrial Engineering from the University of Tehran, Iran in 2010 and 2012, respectively. Her research is in the area of supervisory control of DEDS systems.

Arun Raman is currently working towards the Ph.D. degree in Systems Engineering at the University of Illinois at Urbana-Champaign. He was a Research Associate with the Indian Institute of Management, Ahmedabad from 2014-2015. He worked as an Edison Engineer in GE Oil and Gas in Compressor and Gas Turbine simulation from 2012-2014. He has a Master of Science in Systems and Control from the Indian Institute of Technology Bombay, Mumbai in 2012. His research interests lie in the area of classical control, applied mathematics and the control of DEDS systems.

Ramavarapu Sreenivas received the B.Tech. degree in Electrical Engineering from the Indian Institute of Technology, Madras, India in 1985, and the M.S. and Ph.D. degrees in Electrical and Computer Engineering from Carnegie Mellon University, Pittsburgh, PA in 1987 and 1990, respectively. He was a Postdoctoral Fellow in Decision and Control at the Division of Applied Sciences, Harvard University, Cambridge, MA, before he joined the University of Illinois at Urbana-Champaign in 1992, where he is an Associate Professor of Industrial and Enterprise Systems Engineering.

© IEOM Society International